Claims

1. – 40. (Canceled)

41. (Previously Presented) A computer readable medium storing computer executable instructions for causing a computer system programmed thereby to perform a method of transforming a programming language specification into a lower-level specification, the method comprising:

accepting a programming language specification, the programming language specification including an interface; and

based upon a set of transformation rules, transforming plural methods of the interface into plural ports of a port map of a lower-level specification for a design unit.

42. (Previously Presented) The computer readable medium of claim 41 wherein the method further comprises:

transforming an algorithmic method implementation of the programming language specification into a synchronized process of the lower-level specification.

43. (Previously Presented) The computer readable medium of claim 41 wherein the transforming the interface further includes:

transforming each of zero or more input methods into an input port of the port map; and

transforming each of zero or more output methods into an output port of the port map.

44. (Previously Presented) The computer readable medium of claim 41 wherein a method of the interface includes as a parameter a pointer to a shared variable, wherein the shared variable is for modeling inout behavior.

45. (Previously Presented) The computer readable medium of claim 41 wherein the method further comprises:

transforming plural native programming language data types into hardware description language data types.

46. (Previously Presented) The computer readable medium of claim 41 wherein the programming language specification includes one or more values specified in a template, the programming language specification being based upon the template.

47. (Previously Presented) The computer readable medium of claim 41 wherein the programming language specification is for a system including hardware and software.

48. (Previously Presented) The computer readable medium of claim 41 wherein the programming language specification includes a C++ class description.

49. (Previously Presented) A file including the programming language specification of claim 41.

50. (Previously Presented) A file including the lower-level specification of claim 41.

51. (Previously Presented) A computer readable medium storing computer executable instructions for causing a computer system programmed thereby to perform a method of transforming a programming language specification into a lower-level specification, the method comprising:

accepting a programming language specification, the programming language specification including plural calls to plural instances of a unit class, wherein a first call of the plural calls maps to a first instance of the plural instances of the unit class, wherein a second call of the plural calls maps to a second instance of the plural instances of the unit class, and wherein the programming language specification lacks explicit concurrency modeling for the plural instances of the unit class; and

transforming the programming language specification into a lower-level specification, wherein the transforming includes generating lower-level description for handling concurrent execution of units represented by the plural instances of the unit class in the programming language specification.

52. (Previously Presented) The computer readable medium of claim 51 wherein the programming language specification also lacks synchronization modeling, and wherein the method further comprises generating lower-level description to model synchronization.

53. (Previously Presented) The computer readable medium of claim 51 wherein the method further comprises:

transforming an algorithmic method implementation of the programming language specification into a process of the lower-level specification.

54. (Previously Presented) The computer readable medium of claim 51 wherein the method further comprises:

transforming plural methods of an interface of the programming language specification into plural ports of a port map of the lower-level specification.

55. (Previously Presented) The computer readable medium of claim 51 wherein the programming language specification includes an object-oriented class description.

56. (Previously Presented) A file including the programming language specification of claim 51.

57. (Previously Presented) A file including the lower-level specification of claim 51.

58. (Previously Presented) A design tool comprising:

a design input module for accepting an algorithmic specification, the algorithmic specification including plural unit calls that map to plural different instances of a unit, thereby indicating parallel execution of the plural unit calls; and

a hardware description language transformer for transforming the algorithmic specification into a lower-level specification, wherein the transformer adds code into the lower-level specification for handling the parallel execution of the plural unit calls.

59. (Previously Presented) The design tool of claim 58 further comprising:

an architecture exploration module for presenting alternative architectures for the lower-level specification to a designer.

60. (Previously Presented) The design tool of claim 58 wherein the unit is a sub-design unit of a design unit, and wherein instantiation relationships in the algorithmic specification represent structural relationships within the design unit.

61. (Previously Presented) A file including the algorithmic specification of claim 58.

62. (Previously Presented) A file including the lower-level specification of claim 58.

63. (Previously Presented) A design tool comprising:

one or more modules for accepting an object-oriented description, the object-oriented description including an interface; and

one or more modules for translating the object-oriented description into a lower-level specification having plural ports specified according to defined semantics for the interface of the object-oriented description.

64. (Previously Presented) The design tool of claim 63 wherein the interface includes zero or more input methods, zero or more output methods, and an algorithmic method.

65. (Previously Presented) The design tool of claim 63 wherein the object-oriented description further comprises a second interface comprising one or more declarations, and wherein the second interface is a private interface.

66. (Previously Presented) The design tool of claim 63 further comprising:

one or more modules for presenting alternative architectures to a designer.

67. (Previously Presented) A file including the object-oriented description of claim 63.

68. (Previously Presented) A file including the lower-level specification of claim 63.

69. (Previously Presented) In a computing environment, a computer-implemented method of transforming a programming language specification into a lower-level specification, the method comprising:

providing a programming language specification, the programming language specification including an interface; and

receiving a lower-level specification produced by transforming the programming language specification into the lower-level specification, the lower-level specification having plural ports specified according to defined semantics for the interface of the programming language specification.

70. (Previously Presented) The method of claim 69 wherein the programming language specification is provided to one or more transformer modules, and wherein the lower-level specification is received from the one or more transformer modules.

71. (Previously Presented) The method of claim 69 wherein the computing environment is a distributed computing environment.

72. (Previously Presented) In a computing environment, a computer-implemented method of transforming a programming language specification into a lower-level specification, the method comprising:

providing a programming language specification, the programming language specification including plural unit calls that map to plural different instances of a unit class, thereby indicating parallel execution of the plural unit calls; and

receiving a lower-level specification produced by transforming the programming language specification into the lower-level specification, wherein the transforming includes adding code into the lower-level specification for handling the parallel execution of the plural unit calls.

73. (Previously Presented) The method of claim 72 wherein the programming language specification is provided to one or more transformer modules, and wherein the lower-level specification is received from the one or more transformer modules.

74. (Previously Presented) The method of claim 72 wherein the computing environment is a distributed computing environment.

75. (Previously Presented) A computer-readable medium storing computer-executable instructions for causing a computer system programmed thereby to perform a method comprising:

receiving an object-oriented description, the object-oriented description including native programming language code, wherein the object-oriented description specifies structural details of a design unit for synthesis with a design tool into a HDL description of the design unit; and

compiling the object-oriented description with a native programming language compiler for algorithmic validation of the design unit independent of the synthesis.

76. (Previously Presented) The computer-readable medium of claim 75 wherein the native programming language is C++, and wherein the object-oriented description follows C++ syntax.

77. (Previously Presented) The computer-readable medium of claim 75, wherein the method further comprises:

performing algorithmic simulation with an executable produced by the compiling.

78. (Previously Presented) A file including the object-oriented description of claim 75.

79. (Previously Presented) In a design tool, a method for implementing a hierarchical relationship between a first design unit and a second design unit, the method comprising:

automatically creating one or more local signals of a first design unit, the first design unit including an instance of a second design unit without explicitly specifying connection logic between the one or more local signals and corresponding ports of the second design unit; and

automatically mapping the one or more local signals to the corresponding ports of the second design unit.

80. (Previously Presented) The method of claim 79 further comprising:

before the creating, accepting an algorithmic specification of the first design unit.

81. (Previously Presented) A file including the algorithmic specification of claim 80.

82. (Previously Presented) The method of claim 79 further comprising:

outputting a hardware description language specification for the first design unit.

83. (Previously Presented) A file including the hardware description language specification of claim 82.

84. (Previously Presented) A method of creating a system including software and hardware, the method comprising:

accepting a programming language specification of a system including software and hardware, wherein the programming language specification follows the same syntax for the software and the hardware; and

transforming at least part of the programming language specification into a lower-level specification.

85. (Previously Presented) The method of claim 84 further comprising:

simulating the transformed lower-level specification along with any non-transformed portions of the programming language specification, wherein the simulating co-verifies the software and hardware of the system.

86. (Previously Presented) A file including the programming language specification of claim 84.

87. (Previously Presented) A file including the lower-level specification of claim 84.